Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики и технологий программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

“2D платформер на Unity”

Выполнил

Студент гр. 653502 Новак Д.С

Проверил Козуб В.Н.

Минск, 2017

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ** 3

**1 ПОСТАНОВКА ЗАДАЧИ** 4

**2 ОПИСАНИЕ ПРОГРАММЫ** 5

2.1 ОБЩИЕ СВЕДЕНИЯ 5

2.1.1 ИСТОРИЯ ИГРЫ 5

2.2 ПРАВИЛА И ЦЕЛЬ ИГРЫ 5

2.3 СТРУКТУРА ПРОГРАМММЫ 6

2.4 СХЕМА РАБОТЫ ПРОГРАММЫ 10

**3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 16**

**ЗАКЛЮЧЕНИЕ 17**

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18**

**ВВЕДЕНИЕ**

Игры всегда были интересны людям. Трудно сказать, когда появились первые игры, но совершенно ясно, что они являются неотъемлемой частью жизни человека. Уже в средние века игры были популярны как среди знатных людей, так и среди простого народа. Приводя примеры популярных древних интеллектуальных игр, можно упомянуть шахматы, домино и игру Го.

С развитием технологий игры стали развиваться стремительнее. С созданием первых персональных компьютеров появилась возможность автоматизировать многие игры и создавать реалистичные симуляторы. На сегодняшний день существует целая индустрия, которая предлагает игрокам огромное количество самых разнообразных компьютерных игр, в которую входит большое количество специальностей, по которым работают тысячи человек по всему миру.

Игра «Riley's team» выполнена на Unity.

Unity является современным кросс-платформенным движком для создания двух- и трехмерных игр и приложений. Он позволяет комфортно создавать игры для широкой палитры различных устройств, таких как мобильные устройства, игровые приставки , ПК и других девайсов. Поддерживает различные операционные системы Windows , Linux , OS X, Windows Phone, Android, Apple IO, а также на игровых приставках Wii. PlayStation 3, PlayStation 4 , Xbox One и MotionParallax 3D дисплеях( устройства для воспроизведения виртуальных голограмм), например, Nettlebox.

Стоит отметить основные преимущества работы в Unity. Во-первых то, что в среду разработки интегрирован игровой движок т.е вы можете протестировать не выходя из редактора. Во –вторых, Unity поддерживает импорт огромного количества различных форматов, что позволяет разработчику игры конструировать сами модели в более удобном приложении, а unity использовать напрямую, для разработки продукта. В-третьих, написание скриптов происходит на наиболее популярных языках программирования – C# и JS.

Для своего курсового проекта я выбрала C#. C# является высокоуровневым языком программирования, что позволяет программисту пользоваться различными элементами и приемами, одним словом, преимуществами данного языка.

Таким образом, Unity является актуальной платформой для создания собственных приложений .

Целями и задачами курсового проектирования являются:

\* закрепление знаний среды разработки С#;

\* использование дополнительной литературы;

\* изучение возможностей, предоставляемых Unity;

1. **ПОСТАНОВКА ЗАДАЧИ**

Разработка правил игры:

* Создание иерархии классов для игры, а именно: основной персонаж, несколько видов врагов, декораций, пуль, основной камеры, жизней и монет.
* Подбор спрайтов: для главного персонажа, врагов, а также для фона и декораций.
* Поиск звуковых эффектов для различного рода действий.
* Поиск фоновой музыки
* Создание основного сюжета для игры.
* Создание анимации для персонажей.

При запуске игры на экран будет выведено игровое меню, в котором будет отображаться рекорд игры, кнопка «EXIT» для выхода из игры, а также кнопка «PLAY» после нажатия на которую, произведется переход к главной, игровой сцене.

После перехода к главной, игровой сцене, сверху экрана запускается таймер, который позволит видеть, сколько времени потрачено на прохождение игры, таймер является один из критериев рекорда, а также система подсчета жизней и собранных в течение игры монет. Кроме того, будет предусмотрена система возврата в главное меню, где пользователь сможет изменить настройки игры или выйти из неё.

1. **ОПИСАНИЕ ПРОГРАММЫ**

**2.1 ОБЩИЕ СВЕДЕНИЯ**

Программа была разработана учащейся факультета «ФКСиС» специальности «ИиТП» группы 653502 УО «Белорусский Государственный Университет Информатики и Радиоэлетроники» Новак Дарьей Сергеевной в рамках курсового проекта в 2017 году.

Цель реализации задачи- создание игровой программы, а также знакомство с основными принципами работы в Unity.

Для разработки данной программы был использован язык C# среды Visual Studio 2015, а также Unity 5.5.2. Выбор Unity обусловлен ее кроссплатформенностью, возможностью работать с высокоуровневым языком программирования C#, а также хорошее community, т.е у различных функций движка есть четкое описание с примерами на сайте разработчика, обратиться к которому можно в любой момент, даже без наличия интернета.

**2.2.1 ИСТОРИЯ ИГРЫ**

**Сеттинг**: сказочный мир, наполненный приключениями.

**Об игре**: Мы окунаемся в сказочный мир, наполненный препятствиями. Главный герой - Райли, 11 летний парень, спешит к лучшему другу Кевину, чтобы осуществить их давнюю мечту-сбежать из дома в поисках приключений. Помоги Райли собрать как можно больше денег, чтобы как следует повеселиться.

**2.2 ПРАВИЛА И ЦЕЛЬ ИГРЫ**

Цель игры состоит в том, чтобы главный герой (рис1) за наименьшее время добрался к Кевину (рис 2), собрав при этом как можно больше монет. Время и количество собранных монет фиксируется в системе рекордов, которая позже отображается в главном меню.

**Механики:**

* **Герой:** герой представлен мальчишкой-Райли. Он может ходить, прыгать и стрелять. У героя есть жизни, монетки и волшебные пузыри.

 (рис1) (рис2)

На пути к Кевину главного ждут различные враги и препятствия:

* **Монстры**: приспешники злого монстра, который вызвал их, чтобы они всеми способами мешали нашему герою выполнить свою цель.

 **Красная муха**, преследующая главного героя и желающая нанести урон. Убить муху можно запрыгнув на нее.

 **Передвигающийся враг**, наносящий невероятный урон. Убить его можно запрыгнув на него сверху, а именно между его рожек.

 **Зомби и растенее-убийцы** , стреляющие пулями.

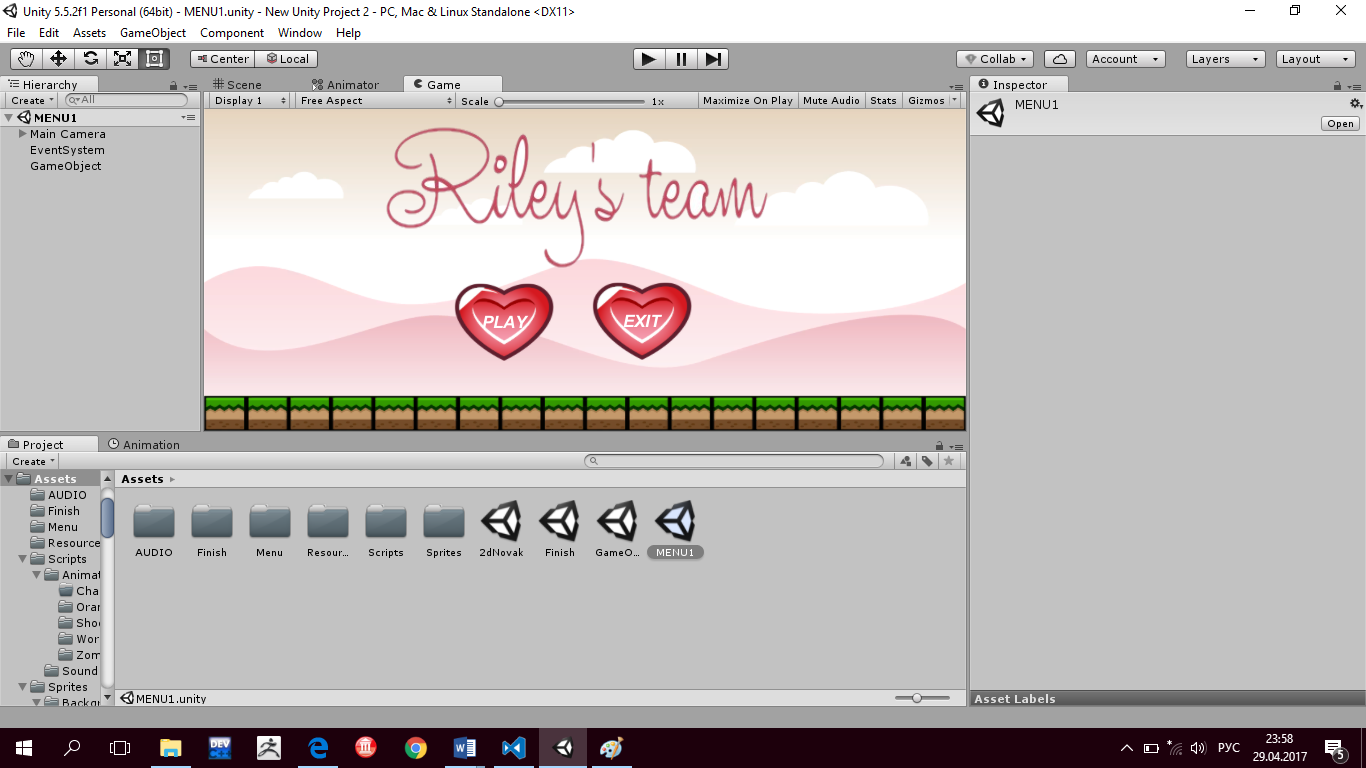
Убить данных врагов можно с помощью пуль главного героя.

 **Монстры –препятствия**, наносящие урон при прикосновении к ним. Убить их невозможно, стоит остерегаться любого прикосновения.

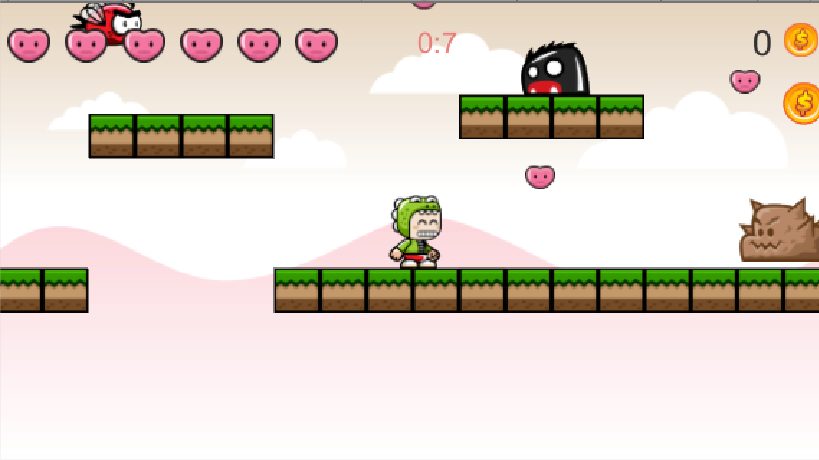
**2.3 СТРУКТУРА ПРОГРАМММЫ**

Алгоритм работы программы.

При запуске игры запускается пользовательское меню(рис1),

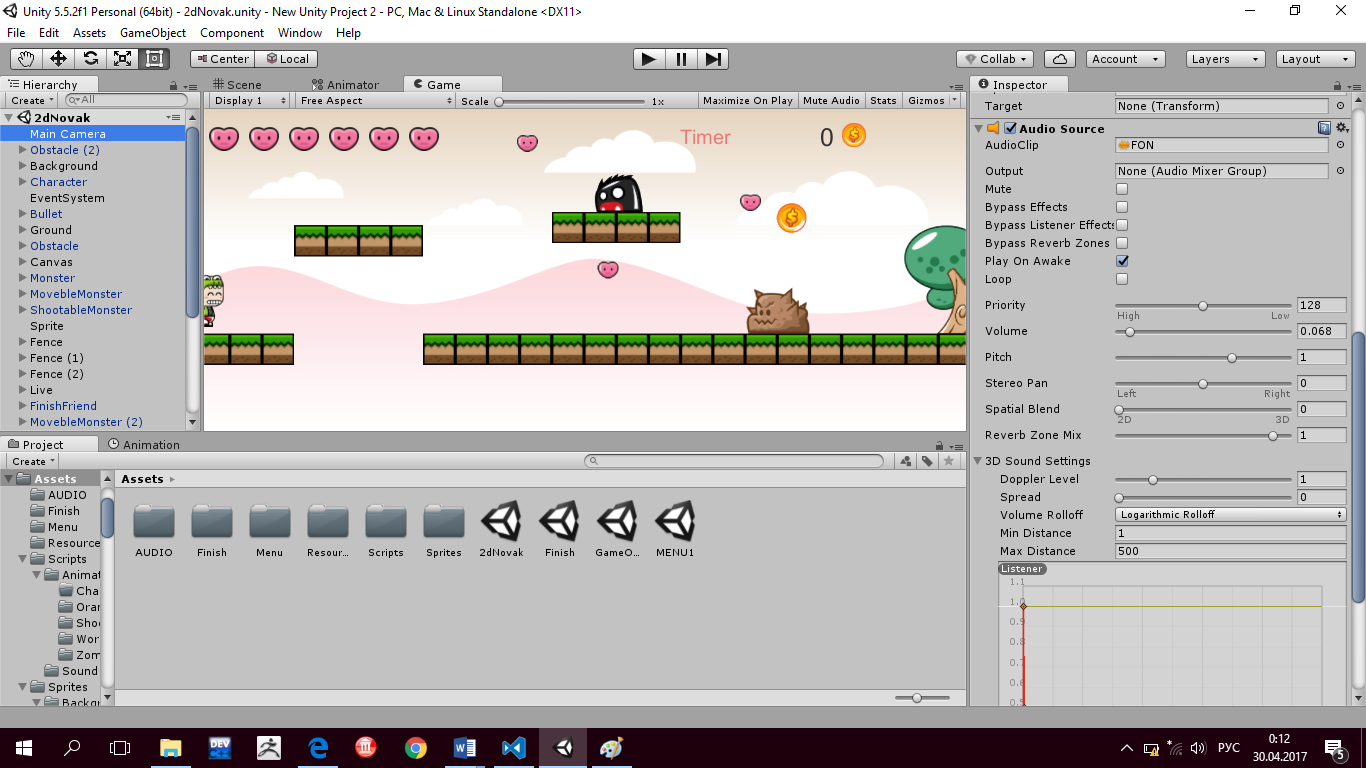
 (рис1)

где будет представлено несколько функциональных кнопок. Кнопка «Play» переходит в главную, игровую сцену(рис2), в которой разворачиваются все действия.

 (рис2)

При старте игры запускается таймер вверху, для фиксирования время прохождения игры и установления последующих рекордов.

Также запускается фоновая музыка во время игры, Music Source прикреплен к основной камере, чтобы мелодия проигрывалась в течение всей игры.(рис 2.2)

(рис 2.2)

В левом углу игры отображается система подсчета жизней главного героя. Жизни могут отнимаются при взаимодействии с монстрами, а также пополняться, при собирании сердечек (рис2)

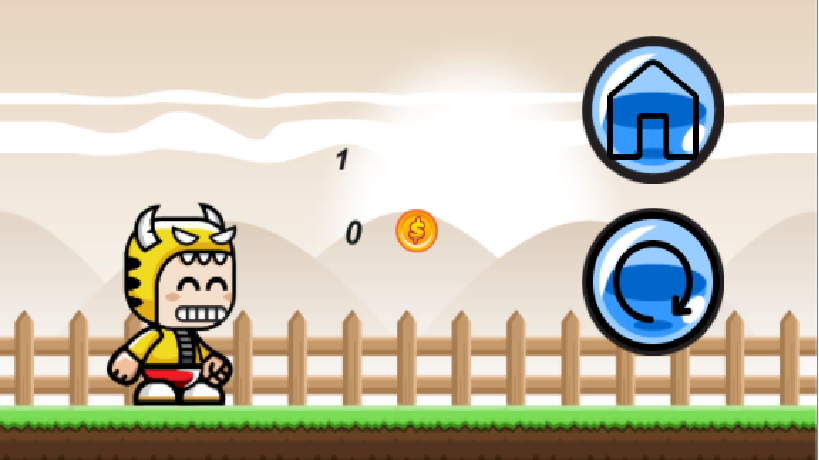
В правом верхнем углу отображаются собранные в течение игры монеты.

Также можно заметить, что в игре присутствуют, так называемые пропасти. Падение в них приводит к мгновенной смерти главного героя и переходу к новой сцене «Game Over». (рис3)

(рис3)

Когда главный герой достигает финиша(рис4), а именно встречается с Кевином, происходит переход в финальную сцену(рис5)

(рис4)

(рис5)

Финальная сцена содержит две кнопки «Home», для возврата в главное меню и «Replay»

Реализация кнопок происходила на Canvas с добавлением Button и прикрепленным к ним скриптом, содержащим переходы в различные сцены.

Также в финальной сцене содержится два текстовых поля, записывающие рекорд игры.

Значение таймера снимается, во время, когда Райли доходит до конца игры, с помощью функции:

private void OnTriggerEnter2D(Collider2D collision)

{

myobj = GameObject.Find("Character");

str = myobj.GetComponent<Timer>().minutes + "min " + myobj.GetComponent<Timer>().seconds + "sec";// функция для получения значения с скрипта Timer в FinishFriend

PlayerPrefs.SetString("GameOverTime", str);

PlayerPrefs.SetString("CoinCount", CoinCounter.instance.CoinCounterBar.text);

SceneManager.LoadScene(2);

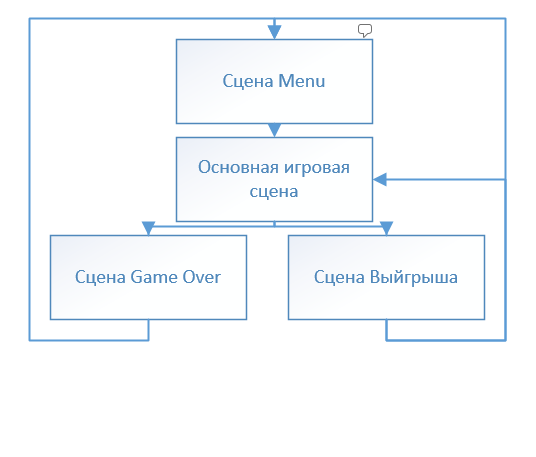
}

А затем, данное значение, записывается в новую сцену, для сохранения результата игры.

Таким образом, в сцене, проигрываемой, когда главный герой дошел до конца, отображается время прохождения игры, а также количество монет, собранных за это время.

**2.4 СХЕМА РАБОТЫ ПРОГРАММЫ**

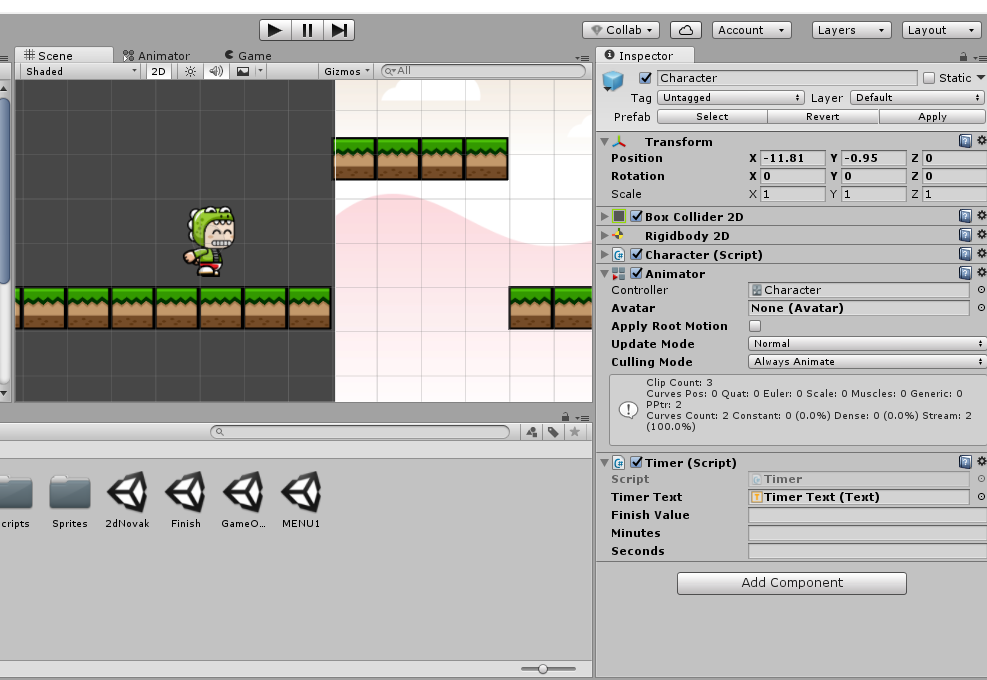
Схема перехода между сцена отображена на блок-схеме(рис1)

(рис1)

Поведение игровых объектов в программе контролируется с помощью компонентов (Components), которые присоединяются к ним. Unity создает свои компоненты, используя скрипты. Они позволяют активировать игровые события, изменять параметры компонентов, и отвечать на ввод пользователя каким вам угодно способом.

В отличии от других ассетов, скрипты обычно создаются непосредственно в Unity.

Так, например, игровой компонент Character(рис2)

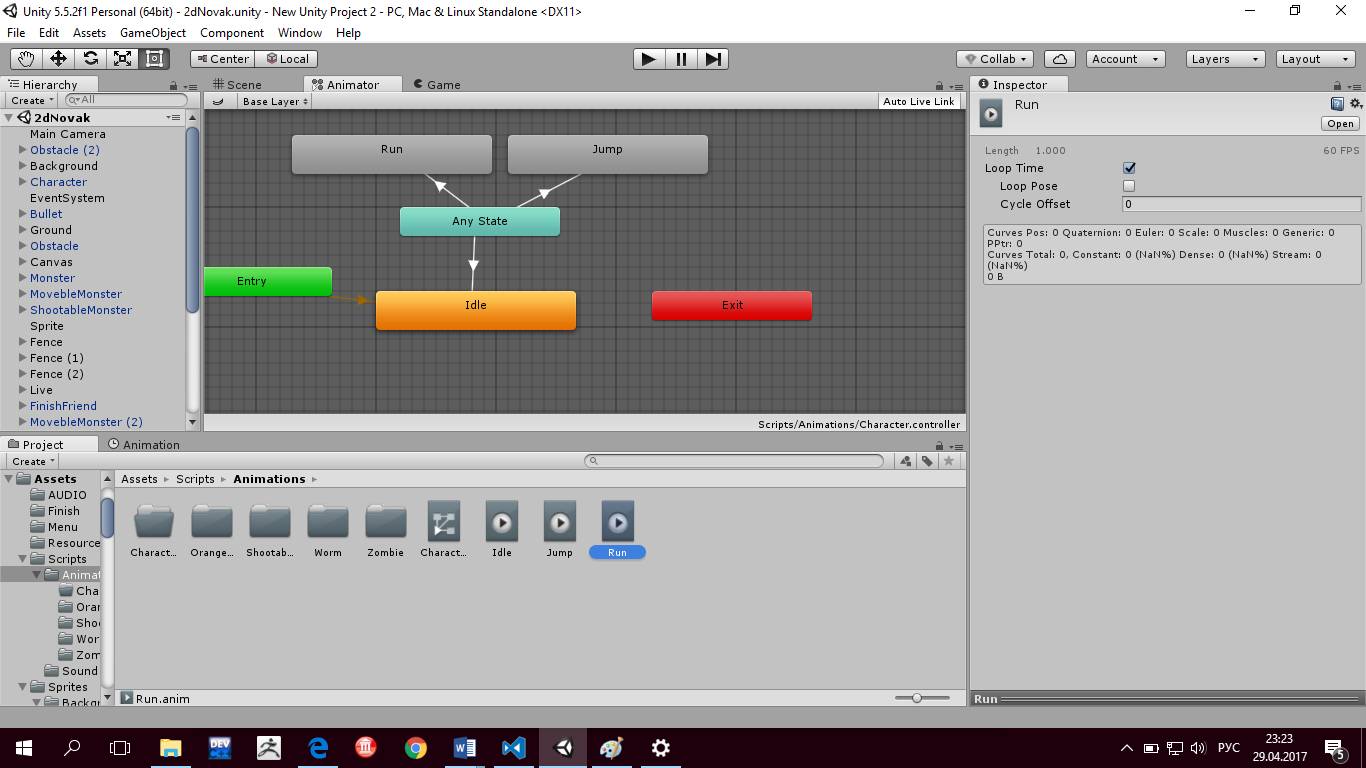
(рис2)

Имеет такие свойства **Rigidbody, Collider, Animator Controller**, а также прикрепленный скрипт Character.

Rigidbody-воплощение физических свойств главного героя. У него есть пара параметров, задающих его взаимодействие с окружающим миром. Собственно, наиболее значимый для динамических объектов: масса. С ней все интуитивно понятно: закон Ньютона и т.д. Приложили силу кратковременно — получили импульс и, соответственно, изменили скорость объекта. Прикладываем длительное время: получаем ускорение в зависимости от массы.

Коллайдер определяет форму главного героя для физических столкновений.

Animator Controller (рис3) также требуется для применения анимаций к скелету. Контроллер руководит набором анимаций для персонажа и переключаться между ними, когда выполняется некоторое условие. Например, главный герой переключается от анимации походки к прыжку при нажатии клавиши пробела. Контроллер управляет переходами между анимациями используя так называемую машину состояний (State Machine)

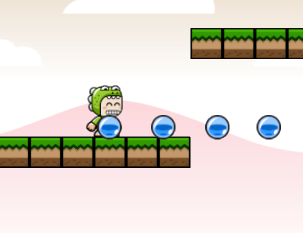
 (рис3)

Что касается самого скрипта, он написан на C# и содержит основные функции для взаимодействия с другими объектами игры.

**Описание алгоритмов для скрипта Character**

Таблица 1 – Описание алгоритмов

|  |  |  |  |
| --- | --- | --- | --- |
| №  п.п. | Наименование  алгоритма | Назначение  Алгоритма | Формальные  параметры |
| 1 | private void Update() | Функция проверяющая состоние героя, вызывается один раз за кадр. Это основное событие для прорисовки кадра. | Возвращаемый параметр: void |
| 2 | private void Run() | Функция отвечающая за перемещение героя, а также за анимацию во время хотьбы. | Возвращаемые параметры: void |
| 3 | private void Jump() | Функция отвечающая за прыжок | Возвращаемый параметр: void |
| 4 | private void Shoot()  (рис4) | Определяет позицию пули относительно главного персонажа(ее координату у, а также ее ориенацию) | Возвращаемый параметр: void |
| 5 | public override void ReceiveDamage() | Отнимает жизни, при прыжке на обьект монст, отбрасывает героя. А также, если жизни закончиись, осуществляет переход в новую сценую | Возвращаемый параметр: void |
| 6 | private void CheckGround() | Проверяет, находится ли герой на земле | Возвращаемый параметр: void |
| 7 | private void OnTriggerEnter2D(Collider2D collision) | Если герой коснулся монстра, отнимает жизни. | Возвращаемый параметр: void |
|  |

(рис4)

На (рис4) отображено, как главный герой стреляет, пули самоуничтожаются в течение секунды.

Скрипт для пули:

using UnityEngine;

using System.Collections;

public class Bullet : MonoBehaviour

{

private GameObject parent;//родитель

public GameObject Parent { set { parent = value; } }

private float speed = 10.0F;

private Vector3 direction;

public Vector3 Direction { set { direction = value; } }

public Color Color

{

set { sprite.color = value; }

}

private SpriteRenderer sprite;

private void Awake()

{

sprite = GetComponentInChildren<SpriteRenderer>();

}

private void Start()

{

Destroy(gameObject, 0.7F);

}

private void Update()

{

transform.position = Vector3.MoveTowards(transform.position, transform.position + direction, speed \* Time.deltaTime);

}

private void OnTriggerEnter2D(Collider2D collider)

{

Unit unit = collider.GetComponent<Unit>();

if (unit && unit.gameObject != parent)

{

if (!(unit is MoveableMonster))unit.ReceiveDamage();

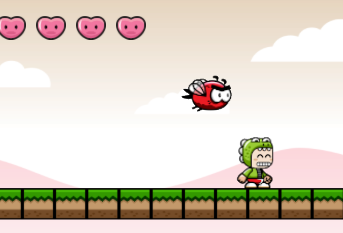
Destroy(gameObject);

}

}

}

Скрипты для монстров содержат уникальные функции, так, например, красная муха(рис5)

(рис5)

обладает, так называемым, разумом, она преследует героя в течение игры(рис4).

Далее представлен класс FlyTOHero для красной мухи.

public class FlyToHero : Monster {

[SerializeField]

new private Rigidbody2D rigidbody;

public float speed = 10f;

private Vector3 pos;

private Vector3 direction;

private SpriteRenderer sprite;

protected override void Awake()

{

sprite = GetComponentInChildren<SpriteRenderer>();

}

protected override void Start()

{

rigidbody = GetComponent<Rigidbody2D>();

direction = rigidbody.transform.right;

}

protected override void Update()

{

Move();

}

protected override void OnTriggerEnter2D(Collider2D collider)

{

Unit unit = collider.GetComponent<Unit>();

if (unit && unit is Character)

{

if (Mathf.Abs(unit.transform.position.x - rigidbody.transform.position.x) < 0.3F) ReceiveDamage();

else unit.ReceiveDamage();

}

}

private void Move()

{

float step = speed \* Time.deltaTime;

pos = GameObject.Find("Character").transform.position;

float MoveToCoor = rigidbody.transform.position.x - pos.x;

if ((rigidbody.transform.position.x > pos.x && MoveToCoor < 5 ) || (rigidbody.transform.position.x < pos.x && MoveToCoor > -5))

{

rigidbody.transform.position = Vector3.MoveTowards(rigidbody.transform.position, pos, step);

}

else

{

Collider2D[] colliders = Physics2D.OverlapCircleAll(rigidbody.transform.position + rigidbody.transform.up \* 0.5F + rigidbody.transform.right \* direction.x \* 0.1F, 0.0F);

if (colliders.Length >= 0)

rigidbody.transform.position = Vector3.MoveTowards(rigidbody.transform.position, rigidbody.transform.position + direction, speed \* Time.deltaTime);

}

}

}

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Разрабатываемый проект является игровой программой "Railey’s Team".

Рекомендуемая конфигурация аппаратного и программного обеспечения:

\* свободного места на жёстком диске – около 150 мегабайт;

\* мышь или сенсорная панель;

\* операционная система семейства Windows.

Для начала работы необходимо запустить исполняемый файл RaileysTeam.exe.

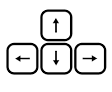
После запуска программы на экране появится главное меню игры.

Управление:

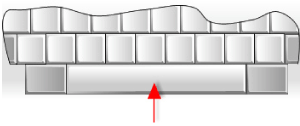
* Левая кнопка мыши: стрельба(рис1)

(рис1)

* Кнопка стрелка вправо, влево: перемещение героя (ходьба)(рис2)

(рис2)

* Кнопка пробел: прыжок(рис3)

 (рис3)

**ЗАКЛЮЧЕНИЕ**

В результате работы над данным курсовым проектом была разработана игровая программа "Сапёр" и документация к ней.

Во время разработки программы были учтены все требования, предъявляемые к программам такого рода: простой интерфейс, удобство в использовании.

В процессе работы над программой были углублены навыки работы с компонентами среды Visual Studio 2015 языка программирования С#, а так же Unity и ее возможности при создании компьютерных игр. Были использованы дополнительные литературные и интернет источники.

Программа является высокоэффективной, компактной и быстрой. Программа отвечает как предъявленным в техническом задании требованиям, так и общепринятым стандартам разработки программного обеспечения такого рода.

**СПИСОК ИСТОЧНИКОВ**

* Unity documentation
* Джозеф Албахари - C# 6.0. Справочник. Полное описание языка
* Unity 3D Game Development by Example (Beginner's Guide)

**Интернет источники:**

* https://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx
* https://habrahabr.ru/